Randomized Local Fast Rerouting for Datacenter Networks with Almost Optimal Congestion

Gregor Bankhamer
Department of Computer Sciences, Universität Salzburg, Austria

Robert Elsässer
Department of Computer Sciences, Universität Salzburg, Austria

Stefan Schmid
TU Berlin, Germany
Faculty of Computer Science, Universität Wien, Austria

Abstract

To ensure high availability, datacenter networks must rely on local fast rerouting mechanisms that allow routers to quickly react to link failures, in a fully decentralized manner. However, configuring these mechanisms to provide a high resilience against multiple failures while avoiding congestion along failover routes is algorithmically challenging, as the rerouting rules can only depend on local failure information and must be defined ahead of time. This paper presents a randomized local fast rerouting algorithm for Clos networks, the predominant datacenter topologies. Given a graph $G = (V, E)$ describing a Clos topology, our algorithm defines local routing rules for each node $v \in V$, which only depend on the packet's destination and are conditioned on the incident link failures. We prove that as long as number of failures at each node does not exceed a certain bound, our algorithm achieves an asymptotically minimal congestion up to polyloglog factors along failover paths. Our lower bounds are developed under some natural routing assumptions.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Distributed algorithms; Networks → Data path algorithms

Keywords and phrases local failover routing, congestion, randomized algorithms, datacenter networks

Digital Object Identifier 10.4230/LIPIcs.DISC.2021.9

Related Version *Full Version*: `https://arxiv.org/abs/2108.02136`

Funding Research supported by the Vienna Science and Technology Fund (WWTF), project ´ICT19-045 (WHATIF), 2020-2024.

1 Introduction

Due to the popularity of data-centric applications and distributed machine learning, datacenter networks have become a critical infrastructure of the digital society. To meet the resulting stringent dependability requirements, datacenter networks implement fast failover mechanisms that enable routers to react to link failures quickly and to reroute flows in a decentralized manner, relying on *static* routing tables which include conditional *local* failover rules. Such local failover mechanisms in the data plane can react to failures orders of magnitudes faster than traditional global mechanisms in the control plane which, upon failure, may recompute routing tables by running the routing protocol again [6, 12, 18].

Configuring fast failover mechanisms however is challenging under multiple link failures, as the failover behavior needs to be pre-defined, before the actual failures are known. In particular, rerouting decisions can only rely on local information, without knowledge of possible further failures downstream. Without precautions, a local failover mechanism may hence entail congestion or even forwarding loops, already under a small number of link failures, while these issues could easily be avoided in a centralized setting.

More formally, resilience is achieved in two stages. First, we are given a graph $G = (V, E)$ describing an undirected network (without failures). Our task is to compute local failover rules for each node $v \in V$ which define for each packet arriving at v to which incident link it should be forwarded, based on the packet's destination (known as destination-based routing); these rules can be conditioned on the status of the links incident to v. Second, when an adversary fails multiple links in the network, packets are forwarded according to our pre-defined conditional rules. Our objective is to define the static routing tables (i.e., the *rulesets*) in the first stage such that desirable properties are preserved in the second stage, in particular, connectivity and a minimal congestion.

This paper studies fast failover algorithms tailored towards Clos topologies, and more specifically to *(multirooted) fat-trees* [1, 17, 22], the predominant datacenter networks. In particular, we consider a scenario where $n-1$ sources inject one indefinite flow each to a single destination. This scenario has already been studied intensively in the literature [2, 4, 11, 21]: it models practically important operations such as in-cast [15, 24], and is also theoretically interesting as it describes a particularly challenging situation because it is focused on a single destination which can lead to bottlenecks.

The goal is to ensure that each flow reaches its destinations even in the presence of a large number of link failures while minimizing congestion: to provide a high availability, it is crucial to avoid a high load (and hence delays and packet loss) on the failover paths. The problem is related to classic load-balancing problems such as balls-into-bins, however, our setting introduces additional dependencies in that failover rules need to define valid paths.

1.1 Our Contribution

This paper studies the theoretical question of how to configure local fast failover rules in Clos topologies such that connectivity is preserved and load is minimized even under a large number of failures.

Results in a Nutshell. We first derive a lower bound showing that by failing $O(n/\log n)$ edges, the adversary can create a load of $\Omega(\log n/\log\log n)$ w.h.p. in arbitrary topologies with n nodes. As a next step, we give a routing protocol for complete bipartite graphs that incorporates local failover rules and, for up to $O(n/\log n)$ link failures, achieves an almost minimal congestion (i.e., up to $(\log\log n)^2$ factor). We then use the derived results to construct a failover ruleset for the Clos topology with $L+1 = \Theta(1)$ levels and degree k (cf the definition in Section 4.1 and the example in Figure 1). It is resilient to $O(k/\log k)$ link failures, while keeping the total load below $O\left(k^{L-1}\log k \cdot \log\log k\right)$ w.h.p. For a certain class of routing protocols, that only forward over shortest paths (according to the local view of the nodes, cf Definition 22) and exhibit a property we call *fairly balanced*, this is again optimal up to $(\log\log n)^2$ factors. This class of protocols is natural and reminiscent of the widely-deployed shortest-path routing protocol ECMP (equal-cost multipath) [16, 22].

Techniques. In this work, we are interested in rulesets that include randomization, and are robust against an adversary, which knows the algorithm and the routing destination, but not the random choices leading to the specific failover routes.

In our lower bound analysis we need to cover a wide variety of failover protocols. While the deterministic case is well-understood [4], and failover protocols based on the uniform distribution are easy to handle, a mixture of the both is non-trivial to analyze. We opt for a carefully crafted case-distinction that captures failover paths, which might be predicted by the adversary with good probability. For this, we exploit the properties of the subgraphs

induced by the edges, which have a certain (high) probability to be chosen as failover links. In all the other cases, we are able to use the (not necessarily uniform) random placement of the loads initiated by a subset of source nodes, and apply a balls-into-bins style argument to show that at least one node will receive high load.

To develop an efficient protocol for the Clos topology, we exploit the fact that it contains multiple bipartite sub-graphs. The main algorithm combines the advantages of deterministic protocols and forwarding loop-freeness, with the resilience of randomized approaches. Our approach builds upon the Interval protocol in [2], which is designed for the clique. However, the adaptation of this approach to the Clos topology comes with multiple challenges that need to be solved. The approach of [2] models the load that nodes receive with the help of trees that are tailored towards the clique, and this method can not be extended to more complex topologies. To overcome this problem, we use a Markov chain to describe such loads and develop a general Markov chain result that might be of independent interest (see Theorem 15). For Markov chains with state space $\mathbb{N}$ that drift towards 0 and that can be modelled with Poisson trials, it states a concentration inequality for the sum of the first $r = \Omega(1)$ elements. Additionally, the protocol in [2] relies on splitting the nodes into partitions of similar size. Contrary to the clique, where the assignment of nodes to partitions can be arbitrary, this is challenging in the case of the Clos topology. Furthermore, for our analysis in the Clos network we need to consider the flows arriving at a certain node from lower and upper levels concurrently. This leads to dependencies, which prevents us from using standard techniques such as Chernoff bounds and the method of bounded differences. To overcome this problem, we uncover the failover edges step by step, and utilize an inductive approach over the increasingly small subtrees around the destination, bounding the number of flows entering the corresponding subtree. For the details see Section 4.3.

1.2 Related Work

Motivated by measurement studies of network-layer failures in datacenters, showing that especially link failures are frequent and disruptive [14], the problem of designing resilient routing mechanisms has received much attention in the literature over the last years, see e.g., [4, 5, 7–10] or the recent survey by Chiesa et al. [6].

In this paper, we focus on the important model in which we do not allow for packet header rewriting or maintain state at the routers, which rules out approaches such as link reversal and others [13, 20]. A price of locality for static rerouting mechanisms has first been shown by Feigenbaum et al. [9] and Borokhovich et al. [4], who proved that it is not always possible to locally reroute packets to their destination even if the underlying network remains connected after the failures. These impossibility results have recently been generalized to entire graph families by Foerster et al. [10]. On the positive side, Chiesa et al. showed that highly resilient failover algorithms can be realized based on arc-disjoint arborescence covers [5, 7, 8], an approach which generalizes traditional solutions based on spanning trees [23]. However, these papers only focus on connectivity and do not consider congestion on the resulting failover paths. Furthermore, while arborescence-based approaches have the advantage that they are naturally loop-free, they result in long paths (and hence likely high load) and are complex to compute.

Only little is known about local fast failover mechanisms that account for load. Pignolet et al. in [21] showed that when only relying on deterministic destination-based failover rules, an adversary can always induce a maximum edge load of $\Omega(\varphi)$ by cleverly failing φ edges; when failover rules can also depend on the source address, an edge load of $\Omega(\sqrt{\varphi})$ can still be generated, when failing φ many edges. In [11], Foerster et al. build upon [4, 21], and leverage

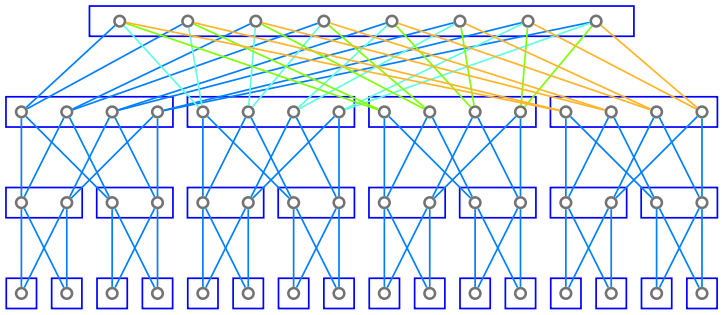

Figure 1 Clos topology with levels $0, 1, 2, 3$ (numbered from top to bottom) and degree $k = 4$.

a connection to distributed computing problems without communication [19], to devise a fast failover algorithm which balances load across arborescences using combinatorial designs. In these papers the focus is on deterministic algorithms.

Our work builds upon [2] where we showed that randomized algorithms can reduce congestion significantly in complete graphs. In particular, we presented three failover strategies: Assuming up to $\varphi = O(n)$ edge failures, the first algorithm ensures that w.h.p. a load of $O(\log n \log\log n)$ will not be exceeded at most nodes, while the remaining $O(\text{polylog}\, n)$ nodes reach a load of at most $O(\text{polylog}\, n)$. The second approach reduces the edge failure resilience to $O(n/\log n)$ but only requires knowledge of the packet destinations, and achieves a congestion of only $O(\log n \log\log n)$ at any node w.h.p. Finally, by assuming that the nodes do have access to $O(\log n)$ shared permutations of V, which are not known to the adversary, the node load can be reduced even further: a maximum load of only $O(\sqrt{\log n})$ occurs at *any* node w.h.p. However, our work relied on the assumption that the underlying network is fully connected (i.e., forms a clique). That said, simulations (performed after we published that paper) also showed promising first results for interval-based routing on Clos topologies.

In this work, we consider randomized fast failover specifically in the context of datacenter networks which typically rely on Clos topologies (also known as multi-rooted fat-trees) [1, 17, 22]. This scenario is not only of practical importance, but also significantly more challenging. Nevertheless, we are able to derive almost tight upper and lower bounds for this setting, under some natural fairness and shortest path assumptions.

1.3 Model

In a nutshell, our model includes two stages. First, we are asked to define the rulesets of the (static) routing tables of each node v in the network; these rules can depend on the destination and be conditioned on the possible link failures incident to v (i.e., only the *local* failures). Later, an adversary will decide which links to fail in the network; as the routing tables defined before are static and cannot be changed depending on the actual failures, packets will now simply be forwarded according to the local failover rules. Our objective is to pre-define these ruleset such that routing reachability is preserved under these failures and the load is minimized.

Local Destination-Based Failover Routing. We represent our network as an undirected graph $\mathcal{G} = (V, E)$ and denote by $\mathcal{F}$ the set of failed edges. Each node v is equipped with a static routing table α, which we assume to be precomputed without knowledge of $\mathcal{F}$. When a packet with destination d arrives at a node v, it is forwarded to the neighbor of v specified in the routing entry $\alpha(v, \mathcal{F}_v, d)$. Here $\mathcal{F}_v = \{w \mid (v, w) \in \mathcal{F}\}$ denotes the set of unreachable neighbors of v (which may be empty). In order to allow for randomization

we assume that, for each node v, the entry $\alpha(v, \mathcal{F}_v, d)$ is drawn from $\mathcal{D}(v, \mathcal{F}_v, d)$, which is a distribution over $(\Gamma(v) \setminus \mathcal{F}_v) \cup \{v\}$. Here $\Gamma(v)$ denotes the set of neighbors of v. This way, a local failover routing protocol $\mathcal{P}$ can be described by the set of its distributions $\mathcal{P} = \{\mathcal{D}(v, \mathcal{F}_v, d) \mid v \in V, \mathcal{F}_v \subseteq \Gamma(v), d \in V\}$. We call such a protocol $\mathcal{P}$ *destination-based* as the only header information that is used for forwarding decisions is the destination address. In the following, we will also assume that for every v the edge (v, v) exists in $\mathcal{G}$ without being included in $\Gamma(v)$. This looping edge cannot be failed by the adversary and is used to enable the analysis of the case where $\mathcal{F}_v = \Gamma(v)$. Throughout the following sections, we assume the existence of an adversary, which knows the employed protocol, or in other words, the set of distributions $\mathcal{P}$ and may construct the set $\mathcal{F}$. However, this adversary does not know the random choices that lead to the routing entries α. In practice, to hide from the adversary longer term, this could be realized by generating new random tables once in a while.

Traffic Pattern and Load. Our focus lies on *flow*-based *all-to-one* routing [11, 21]. That is, we assume that every node $v \in V \setminus \{d\}$ sends out an indefinite flow of packets towards some common destination d. For a node v, we will then say that it has *load* of ℓ (or short: $\mathcal{L}(v) = \ell$) iff ℓ such flows cross node v on their way to destination d. Similarly, for an edge e we define $\mathcal{L}(e)$ to denote the number of flows forwarded over edge e. In case some flow travels in a forwarding cycle, we say that all edges and nodes that lie on this cycle have infinite load.

Because we consider a purely destination-based ruleset, two flows hitting some node w will be forwarded via the same routing entry $\alpha(w, \mathcal{F}_w, d)$. Therefore, as soon as flows hit the same node they cannot be separated anymore. This implies that $\max_{v \in V \setminus \{d\}}\{\mathcal{L}(v)\} = \max_{e \in E}\{\mathcal{L}(e)\}$ as the node v with maximum load in $V \setminus \{d\}$ needs to forward all its flows over the edge (v, d) to reach the destination. As the edge load can be inferred from the node load, we will in the remaining part of our work only consider node loads.

1.4 Conventions and Structure

In the remainder of the paper, when we say we apply Chernoff bounds, we mean the usual multiplicative version. Furthermore, we denote by $\mathrm{Bin}(n, p)$ the binomial distribution with n trials and success probability p, and by $\mathrm{Unif}(A)$ the uniform distribution over elements in the set A. Finally, we denote by w.h.p. ("with high probability") a probability of at least $1 - n^{-\Omega(1)}$, where n is the networks size.

We start by presenting a lower bound in Section 2, stating that there exists a set of edge failures which induces a high load for any network and local destination-based failover protocol $\mathcal{P}$. In Section 3 we present an efficient loop-free failover protocol, which operates in complete bipartite graphs. This is used as a preliminary result to develop a protocol in Section 4, which may be employed in Clos topologies [1]. Each such section comes with a dedicated theorem and the corresponding analysis. Due to space constraints, we refer to the full version of our paper [3] for certain technical details.

2 Lower Bound for Local Destination-Based Failover Protocols

In the following section we construct a lower bound, stating that by failing $O(n/\log n)$ edges the adversary can w.h.p. always create a load of $\Omega(\log n/\log\log n)$.

▶ **Theorem 1.** *Consider any local destination-based failover protocol $\mathcal{P}$ that operates in a graph $\mathcal{G} = (V, E)$ with $|V| = n$ and assume that all nodes perform all-to-one routing to some node $d \in V$. Then, if d and $\mathcal{P}$ are known, a set of failures $\mathcal{F} \subseteq \{(v, d) \mid v \in V\}$ with $|\mathcal{F}| = O(n/\log n)$ can be constructed such that some node $v \neq d$ has $\mathcal{L}(v) > (1/10) \cdot \log n/\log\log n$ w.h.p.*

Most parts of our analysis are concerned with showing that the lower bound in Theorem 1 holds if $\mathcal{G}$ is the complete graph. Therefore, the following notation will be defined with this constraint in mind. We focus on an arbitrary but fixed destination-based failover protocol $\mathcal{P} = \{\mathcal{D}(v, \mathcal{F}_v, d) \mid v, d \in V, \mathcal{F}_v \subseteq V \setminus \{v\}\}$ and also fix the destination node $d \in V$. We will construct a set of failures $\mathcal{F}$ that induces a high load w.h.p. by only failing edges of the form (v, d), i.e., edges incident to the destination. This set of failures will have size $|\mathcal{F}| \leq \varepsilon \cdot n/\log n$, where $\varepsilon > 0$ is an arbitrary small constant. In this setting $\mathcal{F}_v$ – the set of unreachable neighbors of each node v – must either be $\{d\}$ or $\emptyset$ for any node v. We then abbreviate $\alpha(v, \{d\}, d)$ as $\alpha(v)$, which is the node to which v forwards its load in case the link (v, d) is failed. Similarly, we abbreviate the corresponding probability distribution $\mathcal{D}(v, \{d\}, d)$ as $\mathcal{D}_v$ and define f_v to denote the probability density function (PDF) of $\mathcal{D}_v$.

▶ **Definition 2** (Load Graphs). *The following directed graphs lie at the core of our analysis.*

1. $\mathcal{G}_\alpha^\mathcal{F} = (V \setminus \{d\}, E_\alpha^\mathcal{F})$ *where* $E_\alpha^\mathcal{F} = \{(v, \alpha(v, \mathcal{F}_v, d)) \mid v \in V \setminus \{d\}\}$
2. $\mathcal{G}_{\log} = (V \setminus \{d\}, E_{\log})$ *a directed graph with and* $E_{\log} := \{(v, w) \mid v \in V \setminus \{d\} \wedge f_v(w) > 1/\log^4 n\}$
3. $\mathcal{G}_{\log}^t$ *which is constructed from* $\mathcal{G}_{\log}$ *by removing edges in the following way:*
   a. *First, remove arbitrary (outgoing) edges from nodes* v *with* out-deg$(v) > 1$ *until every node has degree* ≤ 1.
   b. *Second, break any remaining cycle by removing an arbitrary edge from each cycle.*

The graph $\mathcal{G}_\alpha^\mathcal{F}$, given a set of failures $\mathcal{F}$ and destination d, describes the path that flows take. In order to fulfill our goal of creating a high load at some node v, we will make sure that some node v is reached by many nodes in $\mathcal{G}_\alpha^\mathcal{F}$. Note that this graph is a random variable as the entries in α follow distributions.

▶ **Observation 3.** *$\mathcal{L}(v) = \infty$ iff v lies on a cycle in $\mathcal{G}_\alpha^\mathcal{F}$. Otherwise $\mathcal{L}(v)$ is equal to the number of nodes $w \in V$ such that a path from w to v exists in $\mathcal{G}_\alpha^\mathcal{F}$.*

The graph $\mathcal{G}_{\log}$ allows us to capture whether the protocol $\mathcal{P}$ contains many failover edges that may be predicted by the adversary. Note, if the edge $(v, w) \in E_{\log}$ and (v, d) are failed, then v forwards its flows to w with probability larger than $1/\log^4 n$. Finally, $\mathcal{G}_{\log}^t$ is just a subgraph of $\mathcal{G}_{\log}$, which does not contain any cycles and simplifies our analysis in some cases. These graphs are related to $\mathcal{G}_\alpha^\mathcal{F}$ in the following way.

▶ **Observation 4.** *If $\mathcal{F}_v = \{d\}$, then $(v, w) \in \mathcal{G}_\alpha^\mathcal{F}$ with probability $f_v(w)$. This probability is independent of other edges (s, t) with $s \neq v$ being in $\mathcal{G}_\alpha^\mathcal{F}$. In case $(v, w) \in \mathcal{G}_{\log}$ (or $(v, w) \in \mathcal{G}_{\log}^t$) it follows that $f_v(w) > 1/\log^4 n$.*

Intuitively, if $\mathcal{G}_{\log}$ contains many edges, then we are in a setting close to determinstic failover protocols. By carefully failing edges of the form (v, d), we have a good chance to make them appear in $\mathcal{G}_\alpha^\mathcal{F}$ and create a node v which is reached by many other nodes. One final definition involves the natural definition of a reverse tree: it is reversed in the sense that all edges are oriented towards the root.

▶ **Definition 5** (Reverse Tree). *We call a directed graph $\mathcal{G}$* reverse tree *iff*

1. *there is a node r in $\mathcal{G}$ with* out-deg$(r) = 0$ *that can be reached from all nodes in $\mathcal{G}$, and*
2. *every node $v \neq r$ in $\mathcal{G}$ has* out-deg$(v) = 1$.

We call r the reverse root *of $\mathcal{G}$. Furthermore, we call a graph $\mathcal{G}'$* reverse subtree *of $\mathcal{G}$ iff $\mathcal{G}'$ is both, a reverse tree and a subgraph of $\mathcal{G}$.*

Note, from the construction of $\mathcal{G}^t_{\log}$ it follows that it is a reverse forest. Let the sets V_R and V^t_R contain the nodes of out-degree 0 in $\mathcal{G}_{\log}$ and $\mathcal{G}^t_{\log}$, respectively. Observation 4 implies for nodes in V_R that we cannot easily predict their forwarding targets. Additionally, when constructing $\mathcal{G}^t_{\log}$ (see Definition 2) in the first step, not a single node has its out-degree modified to 0. This can only happen in the second step, where exactly one node turns into a reverse root. Therefore, the difference $|V^t_R| - |V_R|$ is equal to the number of cycles that were removed in this second step.

Analysis Outline

In the following subsections of the analysis we focus on the complete graph. Depending on the structure of $\mathcal{G}_{\log}$ and $\mathcal{G}^t_{\log}$ we split our analysis into 3 cases. As this graphs are inferred from $\mathcal{P}$, this can also be seen as a distinction between different types of routing protocols. In Section 2.1, we consider the case $|V^t_R| - |V_R| \geq \sqrt{n}$, which intuitively corresponds to the case where $\mathcal{P}$ is prone to produce forwarding loops. In the second case (Section 2.2), we consider $|V_R| \geq \varepsilon n/\log n$, which implies that there are many nodes of degree 0 in $\mathcal{G}_{\log}$. Such nodes do not have a preferred forwarding target in case their link to d is failed. They behave similarly to nodes that forward their flows to neighbors selected uniformly at random. In the last case (Section 2.3), we consider $|V_R| \leq \varepsilon n/\log n$. In this case most nodes have at least one out-going edge in $\mathcal{G}_{\log}$, which can be exploited by the adversary. In any of the three cases, we show that, by failing at most $O(n/\log n)$ edges, a load of $(1/10)\cdot \log n/\log\log n$ is accumulated at some node in the network w.h.p. Finally, we give the proof of Theorem 1 in Section 2.4.

2.1 Analysis Case 1: $|V^t_R| - |V_R| \geq \sqrt{n}$

Recall, the condition of this case implies that $\mathcal{G}_{\log}$ contains at least n^ε many cycles. The idea is, to fail the edge (v, d) for many nodes v that lie on such a cycle. Then, either the whole cycle or at least a long path of nodes lying on such a cycle appears in $\mathcal{G}^{\mathcal{F}}_\alpha$ w.h.p. and causes high load. The detailed proof is given in the full version [3].

▶ **Lemma 6.** *There exists a set of failures $\mathcal{F} \subseteq \{(v,d) \mid v \text{ lies on a cycle in } \mathcal{G}_{\log}\}$ with $|\mathcal{F}| = \sqrt{n}\cdot(1/10)\cdot\log n/\log\log n$ such that some node v that lies on a cycle has $\mathcal{L}(v) \geq (1/10)\cdot\log n/\log\log n$ w.h.p.*

2.2 Analysis Case 2: $|V_R| \geq \varepsilon n/\log n$

In this setting, many nodes v have out-degree 0 in $\mathcal{G}_{\log}$. In case the link (v, d) of such a node is failed, it is hard to predict the failover edge $(v, \alpha(v))$ as these nodes have multiple potential forwarding targets. However, this can be exploited as there must be a set of nodes which are potential forwarding targets of many nodes in V_R. Similarly as in the analysis of a balls-into-bins process, we deduce that, w.h.p., there is one such node that receives load from $\Omega(\log n \cdot \log\log n)$ nodes in V_R. To simplify our analysis, we let V'_R be an arbitrary but fixed subset of V_R with size exactly $\varepsilon n/\log n$. A proof for the following statement is given in the full version [3]. Note that, if the second statement of the following lemma holds, we are already done.

▶ **Lemma 7.** *For $\mathcal{F} = \{(v,d) \mid v \in V'_R\}$ one of the following statements holds:*

1. *In expectation, at least $n^{1/8}$ many nodes $w \in V$ have each at least $(1/10)\cdot\log n/\log\log n$ incident edges that originate from V'_R.*
2. *W.h.p., there exists a node w with $\mathcal{L}(w) = \log^2 n(1-o(1))$.*

The following statement can be shown with the help of standard-techniques, namely the *method of bounded differences* (sometimes also called McDiarmid inequality). This concentration inequality and a detailed proof is given in the full Version [3].

▶ **Lemma 8.** *Let $\mathcal{F} = \{(v, d) \mid v \in V'_R\}$ and assume that the first statement of Lemma 7 holds. Then, w.h.p., there exists a node v such that $\mathcal{L}(v) > (1/10) \cdot \log n / \log\log n$.*

2.3 Analysis Case 3: $|V_R| < \varepsilon n / \log n$

We assume throughout this section that the condition $|V_R^t| - |V_R| \geq \sqrt{n}$ does not hold as that case was already analysed in Section 2.1. This, however, implies that $|V_R^t| < |V_R| + \sqrt{n} < 2\varepsilon n / \log n$. Recall Definition 5 and that $|V_R^t|$ is the number of reverse roots; or in other words, the number of reverse trees in the forest $\mathcal{G}_{\log}^t$. The idea behind this section is simple. If, for the nodes of some reverse subtree in $\mathcal{G}_{\log}^t$ of size $(1/10) \cdot \log n \cdot \log\log n$, we fail the edges incident to destination d, then this whole subtree will appear in $\mathcal{G}_\alpha^{\mathcal{F}}$ with probability $\log^{-(4/10)\log n/\log\log n} n = n^{-4/10}$. By Observation 3 the root of this tree will then receive $(1/10) \cdot \log n / \log\log n$ load. The main challenge is to construct a large enough set of independent trees such that at least one of them appears in $\mathcal{G}_\alpha^{\mathcal{F}}$ w.h.p. This is where the following counting argument comes into play.

▶ **Observation 9.** *If $|V_R| < \varepsilon n / \log n$ and $|V_R^t| - |V_R| < \sqrt{n}$ then $\mathcal{G}_{\log}^t$ must contain one of the following*

1. *$\sqrt{n}$ disjoint reverse trees of size $> (1/10) \cdot \log n / \log\log n$ each, or*
2. *one reverse tree of size $> \sqrt{n}/2$.*

Proof. The proof follows by a counting argument. Assume both statements do not hold. Then, the number of nodes $\mathcal{G}_{\log}^t$ contains can be upper-bounded by

$$\left(\frac{2\varepsilon n}{\log n} - \sqrt{n}\right) \cdot (1/10)\frac{\log n}{\log\log n} + \sqrt{n} \cdot \frac{\sqrt{n}}{2} < n - 1.$$

The first product reflects that all but $\sqrt{n}$ trees have size at most $(1/10) \cdot \log n / \log\log n$. The second product reflects the worst-case of each of these at most $\sqrt{n}$ remaining trees having size $\sqrt{n}/2$. The above inequality chain leads to a contradiction as $\mathcal{G}_{\log}^t$ contains $n - 1$ nodes. ◀

We now present a lemma for both of the cases in Observation 9, each achieving the lower bound in Theorem 1. The proofs follow the ideas sketched at the start of this section. In case of Lemma 11, the tree of size $\geq \sqrt{n}/2$ needs to be split into $\sqrt{n}/\operatorname{polylog} n$ node-disjoint subtrees of size $(1/10) \cdot \log n / \log\log n$. The proofs are given in the full version [3].

▶ **Lemma 10.** *Assume there are $\sqrt{n}$ reverse trees in $\mathcal{G}_{\log}^t$ of size at least $(1/10) \cdot \log n / \log\log n$ each. Then, there exists a failure set $\mathcal{F} \subseteq \{(v, d) \mid v \text{ lies on a tree in } \mathcal{G}_{\log}^t\}$ with $|\mathcal{F}| = \sqrt{n} \cdot (1/10) \cdot \log n / \log\log n$ such that a node v has $\mathcal{L}(v) > (1/10) \cdot \log n / \log\log n$.*

▶ **Lemma 11.** *Assume there is a reverse tree $\mathcal{T}_R$ of size $\sqrt{n}/2$ in $\mathcal{G}_{\log}^t$. Then, there exists a set of failures $\mathcal{F} \subseteq \{(v, d) \mid v \text{ lies in } \mathcal{T}_R\}$ with $|\mathcal{F}| \leq \sqrt{n} \cdot (1/10) \cdot \log n / \log\log n$ such that a node v of $\mathcal{T}_R$ has $\mathcal{L}(v) > (1/10) \cdot \log n / \log\log n$ w.h.p.*

2.4 Proof of Theorem 1

In Sections 2.1–2.3 we considered the complete graph $\mathcal{G}$ together with a fixed destination-based protocol $\mathcal{P}$ and all-to-one destination d. In this setting, we constructed the graphs $\mathcal{G}_{\log}, \mathcal{G}^t_{\log}$ and split our analysis into three cases, depending on the structure of these graphs. In each case, we establish that the theorem's result w.r.t. $\mathcal{G}$ holds:

1. Case 1: If the case in Section 2.1 occurs, then the result immediately follows from Lemma 6.
2. Case 2: If we are in the case of Section 2.2, then either the second statement of Lemma 7 holds and the result follows, otherwise the first statement holds and Lemma 8 leads to the desired result.
3. Case 3: This case was covered in Section 2.3, and further splits into two sub-cases as indicated in Observation 9. In both sub-cases, the result follows as stated in Lemmas 10 and 11, respectively.

The proof for general undirected graphs $\mathcal{G} = (V, E)$ with $|V| = n$ follows from Lemma 26, which is stated in the full version of our paper [3]. The basic idea is that, for any protocol $\mathcal{P}$ operating in $\mathcal{G}$, one can construct an equivalent protocol $\mathcal{P}_K$ that operates in the clique K_n (equivalent in the sense that the path flows take is the same in both graphs). We then use the statement of Theorem 1, which we already established for complete graphs, to deduce that a set of failures $\mathcal{F}^{(K)}$ exists that induces a high load in K_n. The same set of failures (excluding some edges which may not exist in $\mathcal{G}$) also leads to a high load in $\mathcal{G}$ when employing $\mathcal{P}$.

3 Interval Routing in the Bipartite Graph

In the following section, we will construct an efficient local failover protocol for the complete bipartite graph $G = (V \cup W, E)$. Here the set of nodes $V \cup W$ consists of two sets, where $|V| = |W| = n$ and edges are drawn such that each node $v \in V$ is connected to every $w \in W$ and vice versa.

To employ our routing protocol, we further assume that the nodes in both, V and W, are partitioned into $K := C \log n$ sets, where $C = \Theta(1)$ is an arbitrary value larger 4. That is, $V = V(0) \cup V(2) \cup ... \cup V(K-1)$ and $W = W(0) \cup W(2) ... \cup W(K-1)$, where we assume that all these partitions have size $I := n/K = n/(C \log n)$ (assume $C \log n$ divides n). We propose the following local routing protocol, which is resilient to $\Omega(n/\log n)$ edge failures.

▶ **Definition 12** (Bipartite Interval Routing). *We define the routing protocol $\mathcal{P}_B$, induced by the following distributions when routing towards some node $d \in W$*

- *For $v \in V(i)$ we set $\mathcal{D}(v, \mathcal{F}_v, d) = \text{Unif}(\{d\})$ if $d \notin \mathcal{F}_v$, otherwise $\mathcal{D}(v, \mathcal{F}_v, d) = \text{Unif}(W(i) \setminus \mathcal{F}_v)$.*
- *For $w \in W(i)$ with $w \neq d$ we set $\mathcal{D}(w, F_w, d) = \text{Unif}(V((i+1) \mod K) \setminus \mathcal{F}_w)$.*

Note that this protocol is inspired by the *Interval* routing protocol of [2] which is constrained to complete graphs. Intuitively, a packet with source in the set $V(i)$ follows the partitions $V(i) \to W(i) \to V(i+1) \to W(i+1)...$ until reaching a node $v \in V$ such that (v, d) is not failed. Therefore, the only way for flows to end up in a cycle is by travelling through all $2K$ intervals, which is very unlikely. We may also refer to this alternation between layers V and W of a packet as "ping-pong" in the remainder of the paper.

▶ **Theorem 13.** *Let $G = (V \cup W, E)$ be a complete bipartite graph with $|V| = |W| = n$. Let the routing protocol $\mathcal{P}_B$ be employed, configured with $C > 4$, and all-to-one routing towards some destination $d \in W$ be performed. Assume the set of failures $\mathcal{F}$ fulfills for every i with $0 \leq i < K$ that*
1. $\forall w \in W : |\{v \in V(i) \mid w \in \mathcal{F}_v\}| \leq I/3$, *and*
2. $\forall v \in V : |\{w \in W(i) \mid v \in \mathcal{F}_w\}| \leq I/3$.

Then, with probability at least $1 - 3n^{-(C-1)}$, every node $u \in V \cup W$ with $u \neq d$ has $\mathcal{L}(u) = O(\log n \cdot \log\log n)$, even if $\mathcal{F}$ is constructed with knowledge of $\mathcal{P}_B$ and d.

Intuitively, the constraint on $\mathcal{F}$ states that at most a $(1/3)$ fraction of nodes in the same interval may have failed edges incident to the same node. The constraint is, for example, easily fulfilled in case only $I/3 = n/(3C \log n) = \Theta(n/\log n)$ edges are failed in total. This implies that the load induced by the protocol approaches the lower bound in Theorem 1 up to only a polyloglog n factor. In any deterministic protocol, a load of $\Omega(n/\log n)$ could be created in this setting [4]. Additionally, the simple randomized protocol, which forwards the packets between nodes of V and W which are selected uniformly at random until a node $v \in V$ is reached such that (v, d) is not failed, is prone to cycles. By failing $O(n/\log n)$ arbitrary edges between nodes in V and d, at least one flow will travel from such a node $v \in V$ to some $w \in W$ and back to v with probability $\geq 1/\operatorname{polylog} n$. This creates a forwarding loop of length 2 and prevents some flows from reaching destination d. Our interval protocol is hybrid in the sense that nodes forward their packet uniformly at random according to pre-determined partitions. This allows it to keep the network load low while also avoiding forwarding loops w.h.p.

3.1 Analysis of the Bipartite Interval Protocol

We consider a fixed destination node d together with a set of failures $\mathcal{F}$ that fulfills the requirements of Theorem 13. To make our analysis more readable, we assume that all partitions $V(i)$ and $W(j)$ have exactly the same size. Furthermore, we denote by $\alpha(v)$ the (random) node that v forwards packets towards destination d when following $\mathcal{P}_B$. Before starting with the proof of the theorem, we show the following important statement, which implies that, w.h.p., no flows travel in a cycle until they reach the destination d. Packets "ping-pong" between nodes in V and W until they reach the destination. Due to the restrictions on the failure set in Theorem 13 it follows that each time a packet lands on some node $v \in V$, there is a constant probability that the link (v, d) is not failed. It follows that, w.h.p., the packet reaches d after $K = \Theta(\log n)$ alternations between V and W. A detailed proof is given in the full version of our paper [3].

▶ **Lemma 14.** *Any packet starting at some node $u \in V \cup W$ will reach destination d in less than $2K$ hops with probability at least $1 - n^{-C}$.*

The other important ingredient in the proof of Theorem 13 is the following technical statement about Markov chains. A proof for this statement is given in the full version [3]. It exploits that, in expectation, the chain drifts towards 0 with every two further elements.

▶ **Theorem 15** (Markov Chain Aggregation)**.** *Let $\{X_i\}_{i \geq 0}$ be a Markov chain over state space $\mathbb{N}_0$ and $\phi, \psi > 0$ be constants with $\phi \cdot \psi < 1$. Let the following be fulfilled for every $i > 0$:*
1. *X_i can be modeled by a sum of Poisson trials that only depends on X_{i-1}*
2. $\mathbb{E}[X_{2i+1}] \leq X_{2i} \cdot \phi$
3. $\mathbb{E}[X_{2i}] \leq X_{2i-1} \cdot \psi$

Then, there exists a constant $C_{\phi\psi} > 1$, such that for any fixed $r > C_{\phi\psi}$ it holds that $\sum_{i=0}^{r} X_i = O(\log(r) \cdot r)$ with probability at least $1 - 2\exp(-3r)$ as long as $X_0 = O(r)$.

Proof of Theorem 13. We let $V_G := \{v \mid v \in V \wedge (v,d) \notin \mathcal{F}\}$ denote the set of *good* nodes in V that may forward incoming packets directly to destination d. Note that each flow that eventually reaches d, does so over some $v \in V_G$. Therefore, in case no flow traverses a cycle, it follows that the node with maximum load will be some $v \in V_G$. In the following we will consider one such fixed node $r \in V_G$. W.l.o.g. we assume that this node lies in $V(K-1)$ such that we can avoid modulo operations. We define L_j to be the set of nodes whose load r receives within exactly j hops. Clearly $L_0 = \{r\}$, and according to the definition of $\mathcal{P}_B$ it must hold that $L_1 \subseteq W(K-2)$, $L_2 \subseteq V(K-2)$, $L_3 \subseteq W(K-3)$ and so forth. Our goal is to bound the values $|L_j|$ which allows us to determine the load that r receives. To that end, we initially assume that the routing entries $\alpha(v)$ of any node v have not yet been uncovered. Observe that, in order to determine, for example, L_1 it suffices to uncover the entries of nodes in $W(K-2)$ and check which nodes $w \in W(K-2)$ have $\alpha(w) = r$. To determine L_2, we then uncover entries in $V(K-2)$ and check the number of nodes v in this partition having $\alpha(v) \in L_1$. A repetition of this approach step-by-step yields the following intermediate result, which we show in the full version [3] in detail.

▶ **Observation 16.** *The sequence $\{|L_i|\}_{i=0}^{2K}$ forms a Markov chain with $|L_0| = 1$. Additionally, for $i > 0$ it holds that*

1. *$|L_i|$ can be modeled by a sum of Poisson trials depending only on $|L_{i-1}|$*
2. *$\mathbb{E}[|L_{2i+1}|] \leq (3/2) \cdot |L_{2i}|$*
3. *$\mathbb{E}[|L_{2i}|] \leq (1/2) \cdot |L_{2i-1}|$*

Next we make use of Lemma 14. Its statement, together with a union bound application, implies that *no* packet originating from any node travels more than $2K$ hops with probability at least $\geq 1 - |V \cup W| n^{-C} \geq 1 - 2n^{-(C-1)}$. This implies $|L_i| = 0$ for $i \geq 2K$. Hence, our fixed node $r \in V_G$ receives in total $\sum_{i=0}^{2K} |L_i|$ load w.h.p. As the sequence $\{|L_i|\}_{i\geq 0}^{2K}$ is a martingale that follows the properties described in Observation 16, we may apply the Markov chain result Theorem 15 for $r = 2K$. It implies that $\sum_{i=0}^{2K} |L_i| = O(\log n \cdot \log\log n)$ with probability at least $1 - 2n^{-6C}$. Hence, a union bound application yields that for *any* node $r \in V_G$, we have with probability at least $1 - n \cdot (2n^{-(C-1)} - 2n^{-6C}) > 1 - 3n^{-(C-1)}$ that $\mathcal{L}(r) = O(\log n \cdot \log\log n)$. As *no* packet starting at any node $V \cup W$ travels in a cycle, the node with maximum load (excluding d) must be some node $r \in V_G$ and Theorem 13 follows. ◀

3.2 Lower Bound for the Bipartite Graph

In the following, we present a different lower bound variant. It also holds in settings where nodes in W do not contribute one initial flow in the all-to-one routing process. However, it only guarantees high load in expectation as opposed to the high probability guarantee of Theorem 1. We will make use of this version in the analysis of the Clos topology. The proof is given in the full version [3].

▶ **Lemma 17.** *Let $G = (V \cup W, E)$ be a complete bipartite graph with $|V| = |W| = n$ and assume that the nodes in V each initiate one flow towards some node $d \in W$. Then, for any local destination-based failover protocol $\mathcal{P}$, there exists a set of failures $\mathcal{F}$ of size $|\mathcal{F}| \leq \varepsilon \cdot n/\log n$, $\varepsilon > 0$ arbitrary constant, such that, in expectation, the number of nodes with load $\Omega(\log n/\log\log n)$ is at least one.*

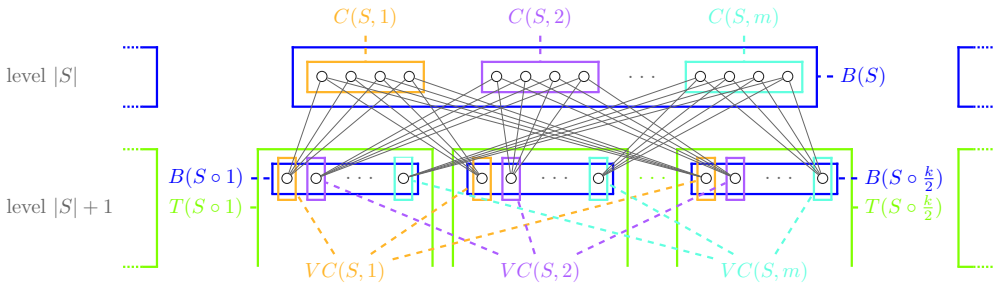

Figure 2 Links between a fixed block $\mathrm{B}(S)$ and its children. Here m denotes the number of clusters in block $\mathrm{B}(S)$.

4 Efficient Protocol for the Clos Topology

4.1 Topology Description

The Clos topology we consider comes with two parameters k, the degree of each node in the network, and $L+1$, $L \geq 1$, the number of levels in the network (cf. also Figure 1). It is constructed as follows. On level 0, there are $(k/2)^L$ many nodes and each level ℓ, $1 \leq \ell \leq L$, consists of $2(k/2)^L$ many nodes. We assume the nodes in each level to be numbered, starting with 1. All nodes are then partitioned into *blocks*. We denote such a block by $\mathrm{B}(S)$, where S is a sequence from the set $\mathbb{S}_L$. This set contains all sequences $S = (s_1, s_2, ..., s_\ell)$ of length $0 \leq \ell \leq L$, where the s_i are integers subject to $s_1 \in [1,k]$ and $s_i \in [1,k/2]$, $i > 1$. The nodes in level ℓ are contained in blocks $\mathrm{B}(S)$ with $S \in \mathbb{S}_L \wedge |S| = \ell$. Each such block contains $(k/2)^{L-\ell}$ many consecutive nodes of level ℓ. In level 0 there is only a single block. In case $\ell > 1$ and $S = (s_1, s_2, ..., s_\ell)$ the block $\mathrm{B}(S)$ contains the nodes $[o+1, o+(k/2)^{L-\ell}]$, where $o = (s_1 - 1) \cdot (k/2)^{L-1} + (s_2 - 1) \cdot (k/2)^{L-2} + \cdots + (s_\ell - 1) \cdot (k/2)^{L-\ell}$.

In the following, we will denote the concatenation operator by $\circ$ and call the blocks $\mathrm{B}(S \circ i)$, $i \in [1, k/2]$, *children* of $\mathrm{B}(S)$ (the block in level $\ell = 0$ has k children) and vice-versa $\mathrm{B}(S)$ the *parent* of the blocks $\mathrm{B}(S \circ i)$. Edges are only drawn between blocks that have a parent-child relationship. This can be seen in Figure 1, where the blocks are visualized as blue boxes (the block at the top is $\mathrm{B}(\emptyset)$). We then denote by $\mathrm{T}(S)$ the subgraph containing all blocks $\mathrm{B}(S')$ such that S is a prefix of S'. For such a fat-tree $\mathrm{T}(S)$, we say that it is *rooted* in $\mathrm{B}(S)$. Note, when compressing each block to a single node and drawing an edge for each parent-child relationship, then the resulting graph becomes a tree such that $\mathrm{B}(S''')$ is a successor of $\mathrm{B}(S'')$ iff S''' is a prefix of S''. In order to describe how edges are drawn, we also define *clusters* such that every block $\mathrm{B}(S)$ is partitioned into clusters. Each cluster $\mathrm{C}(S,i)$, $i \geq 1$, contains the first $i \cdot (k/2)$ consecutive nodes of $\mathrm{B}(S)$. Edges are inserted by constructing complete bipartite subgraphs. For all clusters $\mathrm{C}(S,i)$, we draw edges from every node in the cluster to the i-th node in each of the children of $\mathrm{B}(S)$ (and vice-versa). We call this set of nodes in the children *vertical cluster* $\mathrm{VC}(S,i)$. In Figure 2 we illustrate how these edges are drawn.

Note, from the point-of-view of a fixed node v in some level ℓ, it resides in exactly one cluster of some block $\mathrm{B}(S)$ with $|S| = \ell$. Furthermore, in case of $\ell > 0$, it also lies in exactly one vertical cluster, the cluster $\mathrm{VC}(S', i')$ where $\mathrm{B}(S')$ is the parent of $\mathrm{B}(S)$.

4.2 Routing Protocol

In the following section, we describe how the interval routing protocol of Section 3 can be adapted to the Clos topology. Note that we only consider topologies with a constant amount of layers, i.e., $L = \Theta(1) > 1$. To enable interval routing, we employ an additional layer of

granularity. That is, we partition each cluster and vertical cluster into $K := (4 + L) \log k$ consecutive intervals of size $I := k/((8 + 2L) \log k) = \Theta(k/\log k)$. We denote the j-th such interval, $j \geq 0$, of each cluster $\mathrm{C}(S, i)$, and the vertical cluster $\mathrm{VC}(S, i)$ by $\mathrm{C}(S, i, j)$ or $\mathrm{VC}(S, i, j)$, respectively. A slight exception to this occurs at level 0 which only consists of a single block $\mathrm{B}(\emptyset)$. Here $\emptyset$ is used to denote the sequence of length 0. As $\mathrm{B}(\emptyset)$ has k children instead of $k/2$, there are k nodes in each cluster $\mathrm{VC}(\emptyset, i)$. These vertical clusters are also split into K many intervals, each containing $2 \cdot I$ nodes.

We focus on all-to-one routing towards some destination d which resides on level L (servers are typically located at the bottom of the Clos topology [1]). The primary tool to route packets towards d is the sequence S_d, which we define as the sequence $S \in \mathbb{S}_L$ with length $|S| = L$ that fulfills $\mathrm{B}(S) = \{d\}$. Note that for each node on level L such a sequence must exist (in Figure 1 this is visualized as in the bottom layer each node is contained in its own block). We denote by $S_d|i$, $0 \leq i \leq L$ the length i prefix sequence of S_d. Furthermore, we let $d_{i,j}$ denote the j-th node in the block $\mathrm{B}(S_d|i)$. The routing protocol follows the definitions of a local failover protocol given in Section 1.3 and equips each node with fitting distributions.

▶ **Definition 18** (Clos Interval Routing). *Let v be a node $v \in C(S, i, j)$. Protocol $\mathcal{P}_C$ equips v with the following distributions to enable routing towards d.*

(R1) *S is* not *a prefix of S_d. Let $B(S^P)$ denote the parent of v's block $B(S)$. Then $v \in VC(S^P, i', j')$ for some $i', j' \geq 1$ and*

$$\mathcal{D}(v, \mathcal{F}_v, d) = \mathrm{Unif}\left(C(S^P, i', (j' + 1) \mod K) \setminus \mathcal{F}_v\right)$$

(R2) *S is a length-s prefix of S_d. If $d_{s+1,i} \notin \mathcal{F}_v$, set $\mathcal{D}(v, \mathcal{F}_v, d) = \mathrm{Unif}(\{d_{s+1,i}\})$. Else, set*

$$\mathcal{D}(v, \mathcal{F}_v, d) = \mathrm{Unif}\left(VC(S, i, j) \setminus \mathcal{F}_v\right)$$

The basic idea behind the routing protocol is to send a packets with destination d from child to parent blocks until they reaches a block $\mathrm{B}(S)$ such that S is prefix of S_d (R1). Assume now that, after reaching this block $\mathrm{B}(S)$, the packet lies on a node v_1 in interval $\mathrm{C}(S, i, j)$ and S is a length $|S| = s$ prefix of S_d. As $v_1 \in \mathrm{C}(S, i)$, it is connected to $d_{s+1,i}$, which lies in $\mathrm{VC}(S, i)$. After forwarding the packet to this node, it would then reside on a node in $\mathrm{B}(S_d|s + 1)$. Note that this block's sequence matches the destination for one more element. However, in case $d_{s+1,i}$ cannot be reached, the only link from v into $\mathrm{B}(S_d|s + 1)$ is unreachable. In such a case, it is forwarded to some $w_1 \in \mathrm{VC}(S, i, j)$ instead (R2). As w_1 lies on a block $\mathrm{B}(S')$, which is a child of $\mathrm{B}(S)$ that has some sequence $S' \neq S_d|s + 1$, the packet is forwarded according to R1 in the next step. Afterwards, it will again lie on a node v_2 in $\mathrm{C}(S, i)$. However, this time in the interval $\mathrm{C}(S, i, j + 1)$. In the next step, the packet is again attempted to be forwarded to $d_{s+1,i}$. Otherwise it is forwarded to $\mathrm{VC}(S', i, j + 2)$ and the procedure repeats. Intuitively, the packet "ping-pongs" between layers $|S|$ and $|S| + 1$ until it manages to reach $d_{s+1,i}$, similar as in the protocol for the complete bipartite graph of Section 3. As the forwarding partners are chosen u.a.r., it is unlikely for the packet to hit a node with failed link to $d_{s+1,i}$ in each of $\Omega(\log k)$ alternations, and it will eventually hit $d_{s+1,i}$. A visualization of this idea is given in Figure 3. We also invite the reader to familiarize her- or himself with the more detailed example we prepared in Appendix A.

▶ **Theorem 19.** *Let $G = (V, E)$ be a Clos topology with degree k and $L + 1$ levels for some constant $L > 1$. Consider the routing protocol $\mathcal{P}_C$ and assume all-to-one routing towards some destination d on level L. Assume the adversary chooses its set of failures $\mathcal{F}$ such that the following holds for every triple (S, i, j) where $S \in \mathbb{S}_L$ with $0 \leq |S| \leq L-1$, $1 \leq i \leq L-|S|$ and $0 \leq j < K$:*

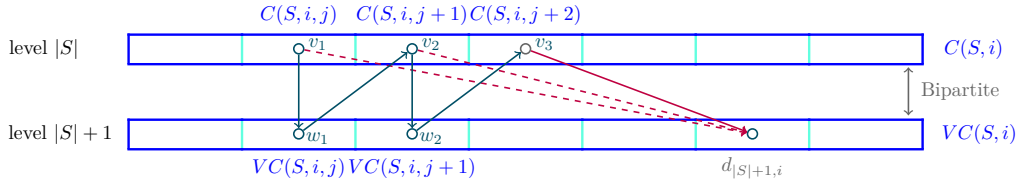

Figure 3 "Ping-Pong" of packet starting at v_1 in a block $\mathrm{B}(S)$ where S is prefix of S_d.

1. $\forall w \in VC(S,i) : |\{v \in C(S,i,j) \mid w \in \mathcal{F}_v\}| \leq I/3$
2. $\forall v \in C(S,i) : |\{w \in VC(S,i,j) \mid v \in \mathcal{F}_w\}| \leq I/3$

Then, with probability $1-O\left(k^{-4}\right)$*, every node* $u \in V \backslash \{d\}$ *has* $\mathcal{L}(u) = O\left(k^{L-1} \cdot \log k \log\log k\right)$*, even if the adversary knows* $\mathcal{P}_C$ *and* d*.*

While the requirement on the failure set $\mathcal{F}$ may seem restrictive at first, it simply states that in every interval at most $I/3 = \Theta(k/\log k)$ many nodes may have failed edges to the same node. Note that $I/3$ failures from nodes of the same interval are simultaneously allowed to many different nodes.

4.3 Analysis of Theorem 19

Throughout this proof, we consider the destination d as well as the set of failed edges placed by the adversary $\mathcal{F}$ to be fixed (we assume this set to adhere to the requirements of Theorem 19). As described in Section 1.3, each node v draws its routing entry $\alpha(v, \mathcal{F}_v, d)$ from $\mathcal{D}(v, \mathcal{F}_v, d)$ which is specified in Definition 18. As we consider the set of failures $\mathcal{F}$ as well as d to be fixed, we use the abbreviation $\alpha(v) := \alpha(v, \mathcal{F}_v, d)$.

Staggered Load Calculation. In the following, we will not immediately uncover all entries $\alpha(v)$ required to determine the load $\mathcal{L}(v)$ some node receives. Instead, we will uncover these entries step-by-step. To that end, we extend our notion of *load* defined in Section 1.3, to also apply in cases where some entries are still left covered. Flows that arrive at a node v with a still covered entry $\alpha(v)$, are assumed to be *stopped* and only contribute to load of nodes that lie on the path the flows takes to reach v. As soon as the entry of v is uncovered, all stopped flows continue to flow until they either reach d, or hit another node with a covered entry. It is easy to see that by increasing the number of uncovered entries, the load at any node can only increase, and, after uncovering all entries of nodes $v \neq d$, we end up with the notion of load defined in Section 1.3. This staggered uncovering of entries allows us to develop a bound on the load step-by-step and helps us circumvent dependencies of the traffic flow in different parts of the topology.

▶ **Lemma 20.** *Let* ℓ *be an integer in* $[0, L-1]$*. Then, after uncovering all entries besides those of nodes in* $T(S_d|\ell)$*, the following holds with probability* $\geq 1 - 4\ell \cdot k^{-4}$*:*

1. *no flow travels in a cycle, i.e.,* $\mathcal{L}(v) < \infty$ *for all nodes* v
2. *flows of nodes with uncovered entries are stopped at some node* $v \in B(S_d|\ell)$
3. *all nodes, including those in* $B(S_d|\ell)$*, have load* $O\left(k^{\ell}\right)$

Sketch of Proof. The proof uses induction over the levels $\ell = 0$ to $L-1$ in the following way. In each step of the induction we uncover the edges in $\mathrm{T}(S_d|\ell) \setminus \mathrm{T}(S_d|\ell+1)$. In the induction hypothesis, we assume that the lemma holds up to some $\ell \in [0, L-2]$ and in the induction step, we show that the statement also holds for $\ell+1$. The base case (i.e., before we uncover any entries) trivially holds.

In order to perform the induction step, we use a two-step approach. First, we uncover the edges in $\mathcal{T}_\ell = \{\mathrm{T}(S_d|\ell \circ i) \mid 1 \leq i \leq k/2 \wedge S_d|\ell \circ i \neq S_d|\ell + 1\}$ (note that if $\ell = 0$, then i is in the range $1, \ldots, k$). As an example, if $\ell = 0$ and d is the last vertex in level 3 in Figure 1, then the set $\mathcal{T}_\ell$ contains the subtrees rooted in the first three blocks of level 1. If $\ell = 1$ (d remains the same node in Figure 1), then $\mathcal{T}_\ell$ is the subtree rooted in the 7th block of level 2. After uncovering the edges in $\mathcal{T}_\ell$, we show that every vertical cluster in the blocks on level $\ell + 1$ in $\mathcal{T}_\ell$ contain $O(k)$ load.

In the second step, we uncover the edges between levels ℓ and $\ell + 1$ in $\mathrm{T}(S_d|\ell)$. Then, we obtain that the statement holds for $\ell + 1$. This second step heavily uses the properties of the failover routing algorithm in complete bipartite graphs. The full proof is given in [3]. ◀

Proof of Theorem 19. Let $S_{L-1} := S_d|L - 1$. We start with an application of Lemma 20 for $\ell = L - 1$. When uncovering all entries except those in $\mathrm{T}(S_{L-1})$, this implies that the flows of all nodes outside $\mathrm{T}(S_{L-1})$ enter $\mathrm{T}(S_{L-1})$ at its root $\mathrm{B}(S_{L-1})$ without causing load higher than $O(k^{L-1})$ w.h.p. Note that the root $\mathrm{B}(S_{L-1})$ of $\mathrm{T}(S_{L-1})$ consists of $k/2$ nodes and therefore only a single cluster. More precisely the following holds.

▶ **Observation 21.** *$T(S_{L-1})$ is a complete bipartite graph that consists of the clusters $C(S_{L-1}, 1)$ and $VC(S_{L-1}, 1)$. These clusters each have size $k/2$ and $d \in VC(S_{L-1}, 1)$.*

This enables us to apply results from the bipartite graph section (Section 3). As S_{L_1} is a prefix of S_d, all flows starting from nodes in $\mathrm{C}(S_{L-1}, 1) \cup \mathrm{VC}(S_{L-1}, 1)$ will "ping-pong" between the clusters until d is reached (see Figure 3). Note that the path taken by the packets in $\mathrm{T}(S_{L-1})$ according to $\mathcal{P}_C$ is exactly the same as if the nodes in $\mathrm{T}(S_{L-1})$ would follow the bipartite routing protocol $\mathcal{P}_B$ instead (described in Section 3 with $C = (4 + L)$). The main result of that section, Theorem 13, then implies that at most $O(\log k \cdot \log\log k)$ load is created w.h.p. However that result assumes that each node in $\mathrm{T}(S_{L-1})$ starts with only 1 flow, while in our case up to $O(k^{L-1})$ flows start from a single node in $\mathrm{C}(S, 1)$ as soon as the entries in $\mathrm{T}(S_{L-1})$ are uncovered. Thus, we obtain a maximum load of $O(k^{L-1} \cdot \log k \log\log k)$. ◀

4.4 Lower Bound for the Clos Topology

Managing load under the all-to-one traffic pattern is inherently challenging even in highly-connected Clos topologies. To illustrate this, we construct a simple congestion lower bound of $\Omega(k^{L-1})$, which holds even in the absence of link failures and does not rely on our notion of local failover routing. Assume that all-to-one routing towards a node d in level L is performed. This destination node d is incident to only $k/2$ many nodes, all of which lie in level $L - 1$. All flows need to travel over one of these $k/2$ nodes to reach d. As the Clos topology contains $\Omega(k^L)$ many nodes in total and each node sends a flow towards d, it follows that one of the $k/2$ many neighbors of d must accumulate $\Omega(k^{L-1})$ flows.

In Appendix B we also present an improved lower bound that only holds for protocols that follow what we call the *faily balanced* and *shortest path* properties (see Definition 22). Among other practically relevant protocols (e.g. ECMP [16, 22]) our protocol fulfills these properties. In this setting, we show that $O(k/\log k)$ edge failures suffice for the adversary to, in expectation, accumulate $\Omega(k^{L-1}\log k/\log\log k)$ load at some node $v \in V \setminus \{d\}$. This implies that the result in Theorem 19 is tight up to a polyloglog factor.

References

1 Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *ACM SIGCOMM computer communication review*, 38(4):63–74, 2008.

2 Gregor Bankhamer, Robert Elsaesser, and Stefan Schmid. Local fast rerouting with low congestion: A randomized approach. In *Proc. 27th IEEE International Conference on Network Protocols (ICNP)*, 2020.

3 Gregor Bankhamer, Robert Elsässer, and Stefan Schmid. Randomized local fast rerouting for datacenter networks with almost optimal congestion, 2021. `arXiv:2108.02136`.

4 Michael Borokhovich and Stefan Schmid. How (not) to shoot in your foot with sdn local fast failover: A load-connectivity tradeoff. In *Proc. International Conference on Principles of Distributed Systems (OPODIS)*, 2013.

5 Marco Chiesa, Andrei V. Gurtov, Aleksander Madry, Slobodan Mitrovic, Ilya Nikolaevskiy, Michael Schapira, and Scott Shenker. On the resiliency of randomized routing against multiple edge failures. In *Proc. ICALP*, 2016.

6 Marco Chiesa, Andrzej Kamisinski, Jacek Rak, Gabor Retvari, and Stefan Schmid. A survey of fast-recovery mechanisms in packet-switched networks. *IEEE Communications Surveys and Tutorials (COMST)*, 2021.

7 Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Andrei Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. On the resiliency of static forwarding tables. *IEEE/ACM Transactions on Networking (TON)*, 25(2):1133–1146, 2017.

8 Marco Chiesa, Ilya Nikolaevskiy, Slobodan Mitrovic, Aurojit Panda, Andrei Gurtov, Aleksander Madry, Michael Schapira, and Scott Shenker. The quest for resilient (static) forwarding tables. In *Proc. IEEE INFOCOM*, 2016.

9 Joan Feigenbaum, Brighten Godfrey, Aurojit Panda, Michael Schapira, Scott Shenker, and Ankit Singla. Brief announcement: On the resilience of routing tables. In *Proc. ACM PODC*, 2012.

10 Klaus-Tycho Foerster, Juho Hirvonen, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. On the feasibility of perfect resilience with local fast failover. In *Proc. SIAM Symposium on Algorithmic Principles of Computer Systems (APOCS)*, 2021.

11 Klaus-Tycho Foerster, Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. Casa: congestion and stretch aware static fast rerouting. In *IEEE INFOCOM 2019-IEEE Conference on Computer Communications*, pages 469–477. IEEE, 2019.

12 Pierre Francois, Clarence Filsfils, John Evans, and Olivier Bonaventure. Achieving sub-second igp convergence in large ip networks. *ACM SIGCOMM Computer Communication Review*, 35(3):35–44, 2005.

13 E.M. Gafni and D.P. Bertsekas. Distributed algorithms for generating loop-free routes in networks with frequently changing topology. *Trans. Commun.*, 29(1):11–18, 1981.

14 Phillipa Gill, Navendu Jain, and Nachiappan Nagappan. Understanding network failures in data centers: measurement, analysis, and implications. In *Proceedings of the ACM SIGCOMM 2011 conference*, pages 350–361, 2011.

15 Mark Handley, Costin Raiciu, Alexandru Agache, Andrei Voinescu, Andrew W Moore, Gianni Antichi, and Marcin Wójcik. Re-architecting datacenter networks and stacks for low latency and high performance. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 29–42, 2017.

16 Abdul Kabbani, Balajee Vamanan, Jahangir Hasan, and Fabien Duchene. Flowbender: Flow-level adaptive routing for improved latency and throughput in datacenter networks. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies*, pages 149–160, 2014.

17 Charles E Leiserson. Fat-trees: universal networks for hardware-efficient supercomputing. *IEEE transactions on Computers*, 100(10):892–901, 1985.

18 Junda Liu, Aurojit Panda, Ankit Singla, Brighten Godfrey, Michael Schapira, and Scott Shenker. Ensuring connectivity via data plane mechanisms. In *10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*, pages 113–126, 2013.

19 Grzegorz Malewicz, Alexander Russell, and Alexander A. Shvartsman. Distributed scheduling for disconnected cooperation. *Distributed Computing*, 18(6):409–420, 2005.

20 Mahmoud Parham, Klaus-Tycho Foerster, Petar Kosic, and Stefan Schmid. Maximally resilient replacement paths for a family of product graphs. In *Proc. OPODIS*, 2020.

21 Yvonne-Anne Pignolet, Stefan Schmid, and Gilles Tredan. Load-optimal local fast rerouting for resilient networks. In *Proc. 47th IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, 2017.

22 Arjun Singh, Joon Ong, Amit Agarwal, Glen Anderson, Ashby Armistead, Roy Bannon, Seb Boving, Gaurav Desai, Bob Felderman, Paulie Germano, et al. Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network. *ACM SIGCOMM computer communication review*, 45(4):183–197, 2015.

23 János Tapolcai. Sufficient conditions for protection routing in ip networks. *Optimization Letters*, 7(4):723–730, 2013.

24 Haitao Wu, Zhenqian Feng, Chuanxiong Guo, and Yongguang Zhang. Ictcp: Incast congestion control for tcp in data-center networks. *IEEE/ACM transactions on networking*, 21(2):345–358, 2012.

A Clos Topology Routing Example

In order to illustrate the behavior of our routing protocol given in Definition 18, we present an exemplary Clos topology in Figure 4. As described in Section 4.1, the nodes are partitioned into blocks, which then are again split into clusters and finally intervals. Note that, in our example, only level $\ell = 0$ has more then one cluster. Additionally, each node on level $\ell = 2$ lies in its own block. These blocks were omitted to improve visual clarity. The goal is to route the packet, currently residing at node s, towards destination node d with sequence $S_d = (3, 6)$. As described in Section 4.1, the nodes are partitioned

First (see a) and b) in Figure 4), the packet is forwarded via R1 to random nodes in an interval of a block that lies one level lower. This is done until the packet resides on a node in $\mathrm{B}(\emptyset)$. This is the first block the packet reaches such that the block's sequence (in this case $\emptyset$) is a prefix of S_d. In the next step, the packet needs to be forwarded via R2 into $T(3)$, the fat subtree containing d, to get closer to the destination d. In our example, we assume that after two hops the random choices caused the packet to land on a node v in cluster $\mathrm{C}(\emptyset, 2)$ of $\mathrm{B}(\emptyset)$. According to the definition of the Clos topology, each node inside this cluster only has a single link into $T(3)$, all of which are incident to $d_{1,2}$. In case no links are failed, the packet would simply be forwarded first to $d_{1,2}$ and then finally towards d. However, in our example we assume that the link $(v, d_{1,2})$ is failed and continue our example with Figure 5.

On the left-hand side in Figure 5 we take a closer look at $\mathrm{C}(\emptyset, 2)$ and its links into level $\ell = 1$. According to the Clos topology definition, the nodes in $\mathrm{C}(\emptyset, 2)$ and the second node in each block $\mathrm{B}(i)$ form a complete bipartite graph. These "second nodes" (denoted by $\mathrm{VC}(\emptyset, 2)$) are partitioned into vertical intervals, separated by the orange lines in the image. The idea is now to forward the packet, currently residing on v, towards $d_{1,2}$ just as in the protocol for the bipartite graph in Section 3. That is, until the packet reaches a node v' such that the link to $d_{1,2}$ is not failed, it "ping-pongs" between intervals of $\mathrm{C}(\emptyset, 2)$ and vertical intervals of $\mathrm{VC}(\emptyset, 2)$. Our routing protocol implements this as the forwarding rule applied to the packet alternates between R2 and R1.

After being forwarded to $d_{1,2}$, we continue our example with the right-hand side of Figure 5. The packet now resides in $T(3)$ and is close to the destination. If the link $(d_{1,2}, d)$ is not failed, then the packet is forwarded directly to d via R2. However, we assume that

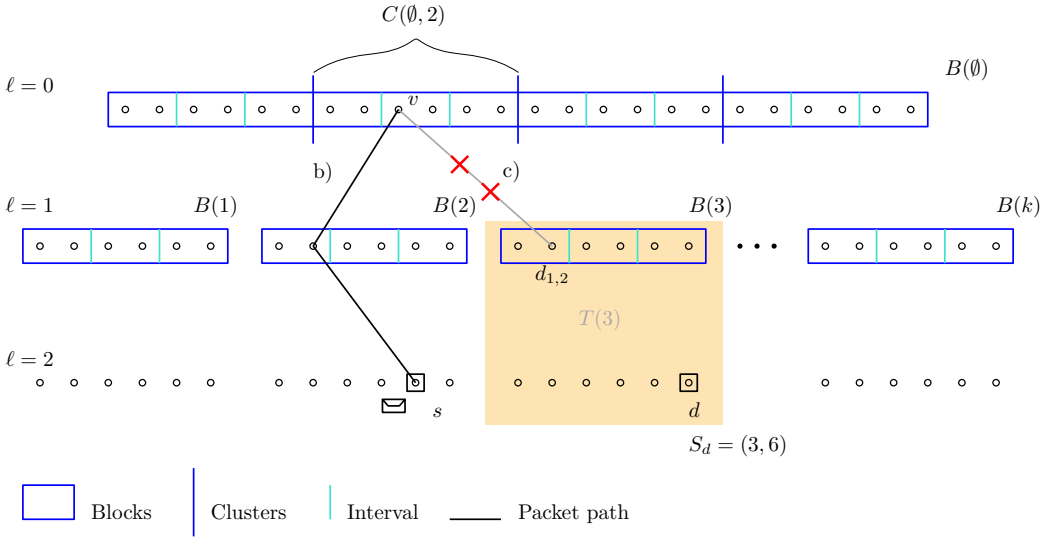

Figure 4 Clos topology with $k = 12$, $L = 2$ and interval size $I = 2$. For easier visibility, edges are not visible. A packet residing at some bottom node of layer s is forwarded towards destination d with sequence $S_d = (3, 6)$ until it encounters a failed edge.

this is not the case. Just as in subgraph considered in the left-hand side of Figure 5 one can again observe that $T(3)$ is a complete bipartite graph. This allows us to again apply ideas from the bipartite routing protocol. The packet "ping-pongs" via R1 and R2. until it hits the first node in $B(3)$ that can reach d directly.

B Refined Clos Lower Bound

In this section, we present an improved lower bound that is targeted towards a class of routing protocols that exhibit the following properties.

▶ **Definition 22** (Fairly Balanced and Shortest Path Routing). *Let $\mathcal{P}$ be a local failover protocol operating in the Clos topology, assume $\mathcal{F} = \emptyset$, i.e., no edges are failed, and assume that all-to-one routing towards any arbitrary destination d on level L is performed. We call $\mathcal{P}$ a* fairly balanced *protocol if, w.h.p., it holds that $\mathcal{L}(v) = \Theta(k^\ell)$ for $v \in B(S_d|\ell), 1 \leq \ell \leq L$, and $\mathcal{L}(v) = O(\text{polylog}\, k)$ otherwise.*
Furthermore, we say $\mathcal{P}$ is a shortest path *protocol, or $\mathcal{P}$* forwards over shortest paths *if the following holds for every node $v \in V \setminus \{d\}$. Let $\mathcal{F}$ be a set of failures (which might be empty) such that $|\mathcal{F}_v| \leq I/3 = O(k/\log k)$. Then, the routing entry $\alpha(v, \mathcal{F}_v, d)$ must always lie on a shortest path to d in the graph $(V, E \setminus \mathcal{F}_v)$.*

While assuming these properties limits the generality of our lower bound, they are natural and realized by the standard equal-cost multipath protocol ECMP [16,22] which also underlies the widely-used routing protocols OSPF.[1] Intuitively, fairly balanced means that the load that has to enter some block at a particular level of the Clos topology is "fairly" balanced among the nodes of the particular block. That is, each such node receives the same load up to constant factors. Protocols, which do not exhibit the fairly balanced property, seem unnatural as they may generate load situations in which some nodes are heavily affected by flows while others (on the shortest path from a level 0-node to the destination) remain idle.

[1] Specifically, ECMP balances flows across shortest paths by default, and upon a failure, locally re-hashes to redistribute flows across the remaining shortest paths to the destination.

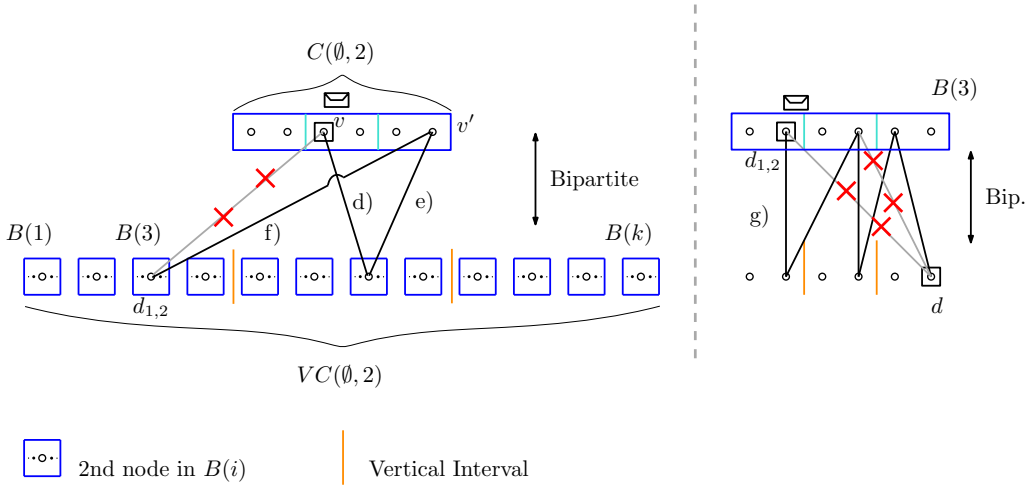

Figure 5 On the left, a closer look at $\mathrm{C}(\emptyset, 2)$. The nodes in $\mathrm{C}(\emptyset, 2)$ form a complete bipartite graph together with the second node in each block $\mathrm{B}(i)$. On the right, we have $T(3)$. After landing on $d_{1,2}$, the packet cannot be forwarded directly to d as we assume the link $(d_{1,2}, d)$ is failed.

However, note that ECMP protocol may generate cycles with probability $1/\operatorname{polylog} n$ if the number of failures is $\Omega(k/\log k)$. For protocols that exhibit above properties, one can construct a load lower bound of $\Omega(k^{L-1}\log n/\log\log n)$. To achieve this bound, the adversary only fails edges in $\mathrm{T}(S_d|L-1)$, which is a complete bipartite graph (see Observation 21) consisting of k nodes partitioned into $\mathrm{C}(S_d|L-1,1)$ and $\mathrm{VC}(S_d|L-1,1)$. As we only consider fairly balanced protocols it follows that each node in $\mathrm{C}(S_d|L-1,1)$ (the one partition of the complete bipartite graph) receives a load of $\Theta(k^{L-1})$.

Due to shortest path routing, no load will ever leave this bipartite graph again. Then, a set of edges incident to the destination can be failed such that a load of $\Omega(k^{L-1}\log k/\log\log k)$ is generated. The corresponding results can be found in Lemma 17 of Section 3.2. We then show in Lemma 24 that our routing protocol $\mathcal{P}_C$ indeed fulfills the properties in Definition 22. This implies that the result in Theorem 19 is tight. The proofs are given in the full version [3].

▶ **Lemma 23.** *Let $\mathcal{P}$ by a fairly balanced protocol that operates in a Clos topology with $L = \Theta(1) > 1$ layers and only forwards over shortest paths to some destination d on level L. Then, there exists a set of failures $\mathcal{F}$ with $|\mathcal{F}| \leq I/3 = O(k/\log k)$ such that, in expectation, at least one node $v \neq d$ has $\mathcal{L}(v) = \Omega(k^{L-1}\log k/\log\log k)$.*

▶ **Lemma 24.** *The protocol $\mathcal{P}_C$ defined in Definition 18 is a fairly balanced and shortest path protocol.*